**1050 Programming Logic**Object-Oriented Programming – Inheritance Project

NAME: **Angela Dennis**

***Instructions:*** *Complete the following exercises. Push your code to GitHub and share the URL to your repository by submitting it to Blackboard.*

* Do Exercise 11.9 (Account Inheritance Hierarchy) [ page 466 of Chapter 11 of text book ]
* Point distribution will closely follow 11.9 and is as follows if the submitted program is compilable and runnable:
* Base Class – 5 points
* Derived Class One – 5 points
* Derived Class One – 5 points
* Method Credit – 5 points
* Method Debit – 5 points
* Method CalculateInterest -5 points
* Redefined methods in CheckingAccount – 10 points
* Main method with tests – 10 points
* Total – 50 points.

namespace Lab11

{

public class Account

{

decimal balance;

public decimal Balance

{

get { return balance; }

set

{

if (value >= 0.0M)

{

balance = value;

}

else

{

balance = 0.0M;

Console.WriteLine("Initial balance was invalid, zero used.");

}

}

}

public Account(decimal balance)

{

Balance = balance;

}

public virtual void Credit(decimal amount)

{

balance += amount;

}

public virtual bool Debit(decimal amount)

{

if (amount <= balance)

{

balance -= amount;

return true;

}

else

{

Console.WriteLine("Debit amount exceeded account balance.");

return false;

}

}

}

public class SavingsAccount : Account

{

decimal interestRate;

public SavingsAccount(decimal balance, decimal interestRate) : base(balance)

{

this.interestRate = interestRate;

}

public decimal CalculateInterest()

{

return Balance \* interestRate / 100m;

}

}

public class CheckingAccount : Account

{

decimal fee;

public CheckingAccount(decimal balance, decimal fee) : base(balance)

{

this.fee = fee;

}

public override void Credit(decimal amount)

{

base.Credit(amount);

Balance -= fee;

}

public override bool Debit(decimal amount)

{

if (base.Debit(amount))

{

Balance -= fee;

return true;

}

else

{

return false;

}

}

}

class Program

{

static void Main()

{

Console.Write("Welcome to Tri-C Bank: ");

Console.Write("\n \n");

//This is a test to show how the error would happen by inputting numbers.

Account acc1 = new Account(100);

Console.WriteLine("The intial balance of your account is {0:C}", acc1.Balance);

acc1.Credit(100);

acc1.Debit(50);

Console.WriteLine("The final balance of your account is {0:C}", acc1.Balance);

Console.Write("\n");

Account acc2 = new Account(-100);

Console.WriteLine("The intial balance of your account is {0:C}", acc2.Balance);

acc2.Credit(30);

acc2.Debit(40);

Console.WriteLine("The final balance of your account is {0:C}", acc2.Balance);

Console.WriteLine();

//This is simulating a real bank acct, allowing amounts to be put in and out

SavingsAccount acc3 = new SavingsAccount(100, 5);

Console.WriteLine("The intial balance of your Savings account is {0:C}", acc3.Balance);

Console.Write("How much are you depositing? ");

int operand1 = Convert.ToInt32(Console.ReadLine());

Console.Write("How much are you taking out? ");

int operand2 = Convert.ToInt32(Console.ReadLine());

acc3.Credit(operand1);

acc3.Debit(operand2);

Console.WriteLine("The current your Savings account is is {0:C}", acc3.Balance);

decimal interest = acc3.CalculateInterest();

acc3.Credit(interest);

Console.WriteLine("With Interest, the final balance of your Savings account is {0:C}", acc3.Balance);

Console.WriteLine();

CheckingAccount acc4 = new CheckingAccount(100, 2.5M);

Console.WriteLine("The intial balance of your Checking account is {0:C}", acc4.Balance);

Console.Write("How much are you depositing? ");

int operand3 = Convert.ToInt32(Console.ReadLine());

Console.Write("How much are you taking out? ");

int operand4 = Convert.ToInt32(Console.ReadLine());

acc4.Credit(operand3);

acc4.Debit(operand4);

Console.WriteLine("With Interest, the final balance of your Checking account is {0:C}", acc4.Balance);

Console.ReadKey();

}

}

}

